

# Algorithms (2023 Summer)

#1: イントロダクション,  
計算量

矢谷 浩司

# ご登録をお願いします！

本学学生：<https://forms.gle/eYiZRVPfxM1qx5A9>

メタバース工学部生：（メールでお願いしているフォーム）  
今すぐお願いします！

講義のslackやコードチャレンジで使うシステムへの招待を行う他，皆さんのバックグラウンドを把握するため，ご協力をお願いします。

単位の取得が必要な人は，別途UTASでの登録を忘れないようにお願いします。

# Welcome!

この講義を担当する矢谷浩司です。

今学期どうぞよろしくお願いいたします！

アルゴリズムとは？

# アルゴリズムとは

あるタスクを達成するために設計された有限回の計算手順（ソート，サーチ，最適化などなど）。

アルゴリズムは正しい，あるいは「最適な」解を導くように設計されている。（ただし，最適と言っても，あくまでアルゴリズム内で設定された評価基準による）。

多くの場合は（時間コスト，メモリコスト，通信コストにおいて）効率的な計算手順を意味する。

# アルゴリズムの例

文章の中から所望の単語がある場所を探し出す。

たくさんの数字を小さいもの順に並べる。

A駅からB駅に行くまでの電車でのルートを検索し、  
運賃の安い順に並べる。

進学選択で各学生さんの配属学科を決める。

# アルゴリズムがわかると

効率的な処理を設計できる。

与えられた処理がどの程度計算を要するものなのかを見積もることができる。

ボトルネックになっているコードを解析してそこを改善できる。

# この講義でやること

データ構造やアルゴリズムに関する基礎知識の学習

それらをpythonで実装し、実際に体験

さらにそれらを利用して、演習課題に取り組む



この講義で学んでほしいこと:

頭で考えた処理手順をコード  
に落とし込む

この講義を終えた後にはこうなってほしい

データ構造やアルゴリズムに関する基礎的な話がわかっている。

それらをpythonで実装することができる。

それらを応用するような課題に自分で取り組める。  
(例えば、競技プログラミングなど)

この講義のモットー:

Let's code! 😄

# 講義内容を設計するにあたって

アルゴリズムを初めて学ぶ人が大部分， という前提です。

コーディングを得意としない人にも理解してもらえるように， かなり噛み砕いてスライドを作ったつもりです。

また， Let's code!のモットーの下， 毎週手を動かす機会を提供するようにしています。

# この講義の対象者

アルゴリズムを初めて学ぶ（もしくはそれに近い）。  
勉強してみたが、挫折した人もvery welcome!

プログラミング，Pythonの基本は一応理解している。

競技プログラミングとか，ちょっと始めてみたいかも，  
と思っている。

この講義の対象者でない方 😊💧

「蟻本読破しました。」


「暖色コーダーです。」

「プログラミングコンテスト入賞者です。」

上記方々はこの講義を受けても多分非常につまらんと  
思いますので、ぜひ我が道を行ってください！

この講義が（積極的に）カバーしないこと

アルゴリズムの正当性の証明  
計算量の細かい議論，証明  
数学的理論

この講義は，アルゴリズム・データ構造を直感的に理解して，手を動かすことを重視するスタイルですので，理論や説明の厳密性には目を瞑ってあげてください。  
（説明は丁寧に行うよう心がけています。）

ちなみに、

残念ながら、私はアルゴリズムの専門家ではありません。

専門はHCI（ユーザインタフェース）

<https://iis-lab.org>, <https://yatani.jp>

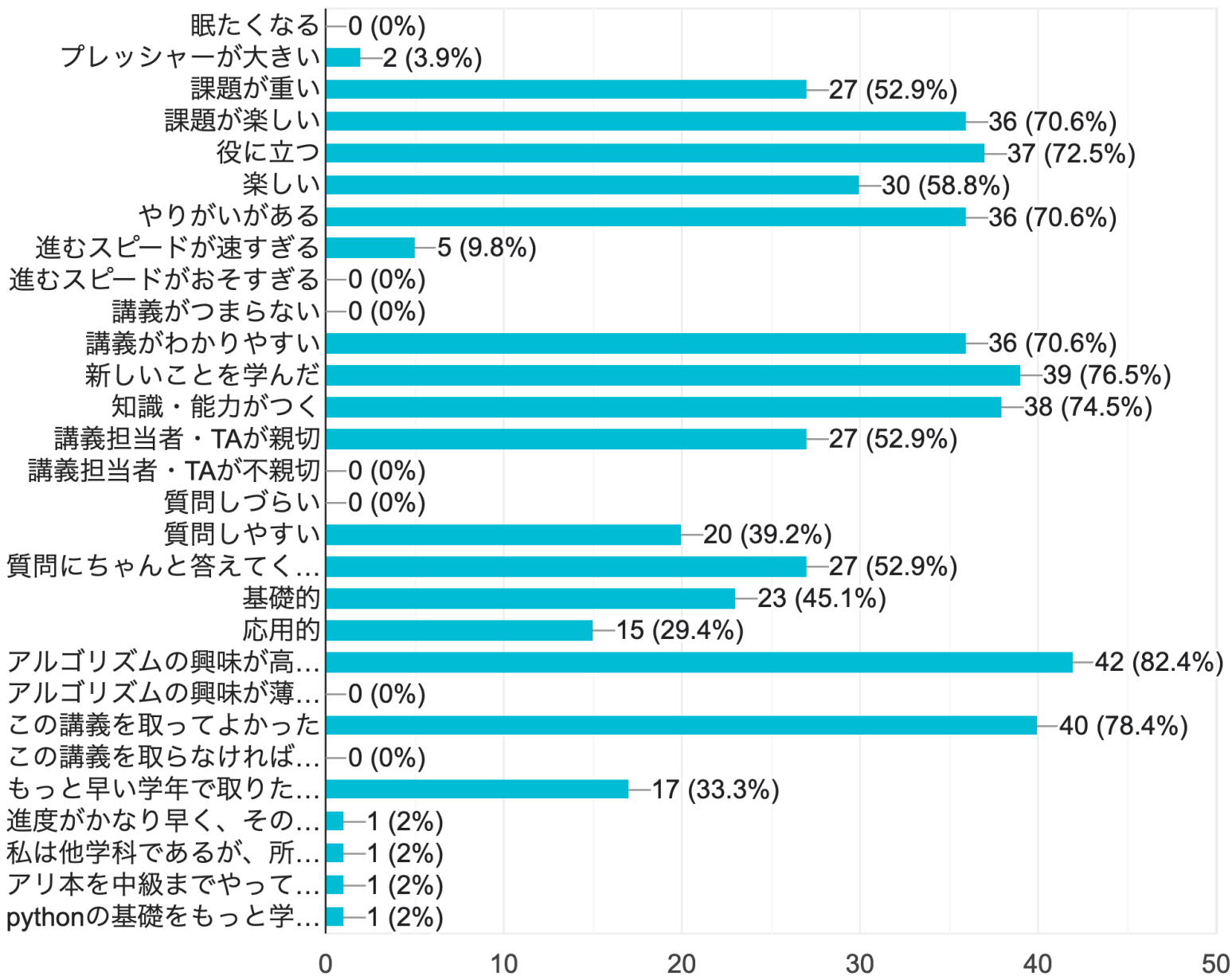
ですが、情報技術や情報を人にとってわかりやすい形に設計することには、強い興味&自信があります。

この講義でも「わかりやすさ」を重視してご説明を差し上げていきたいと思っておりますので、どうぞお付き合いください。



この講義での体験を他の人に伝えるときに使うで...う言葉を選んでください（当てはまるもの全て）

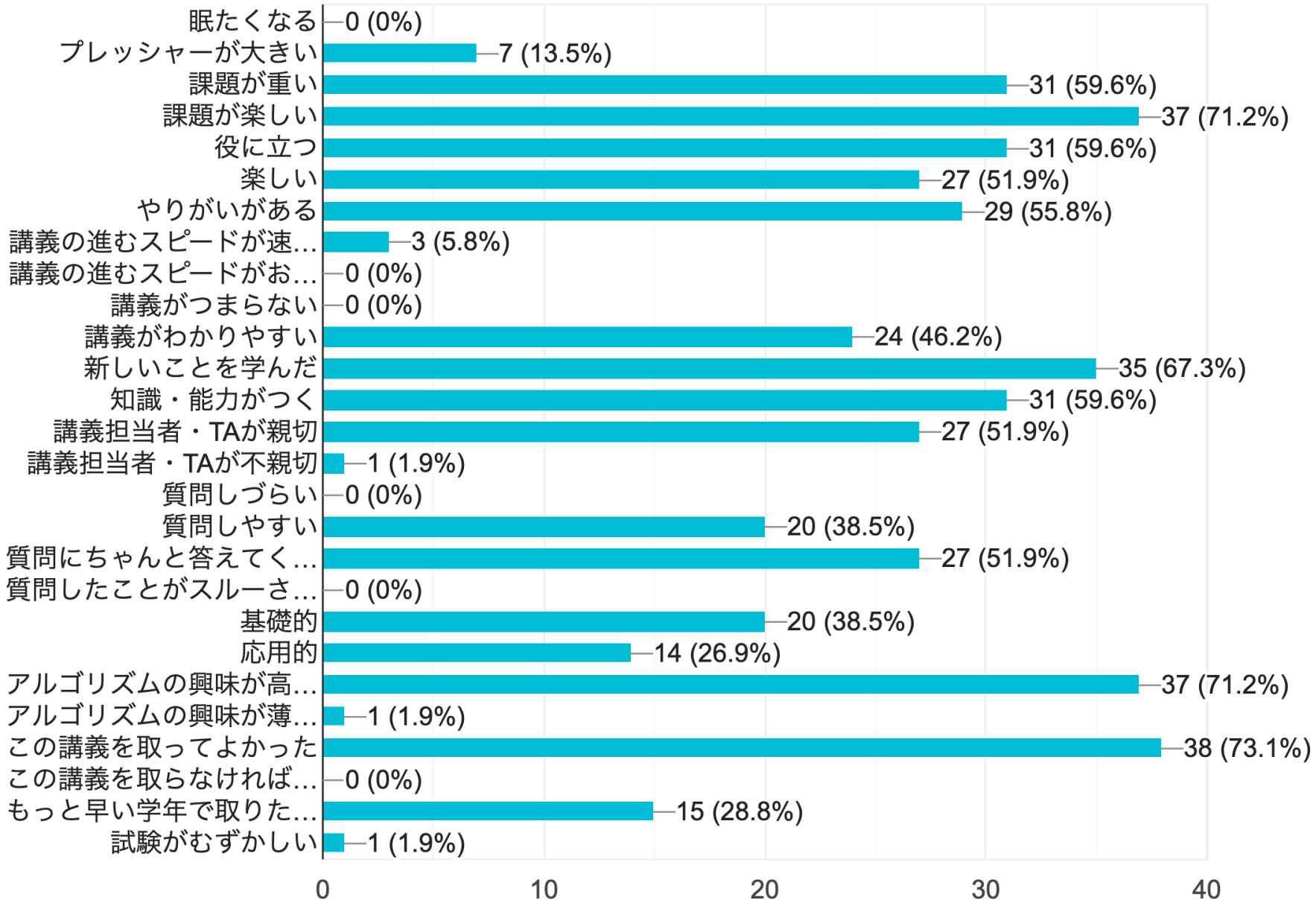
51 responses



2020年度  
アンケート結果

この講義での体験を他の人に伝えるときに使うであ...う言葉を選んでください（当てはまるもの全て）

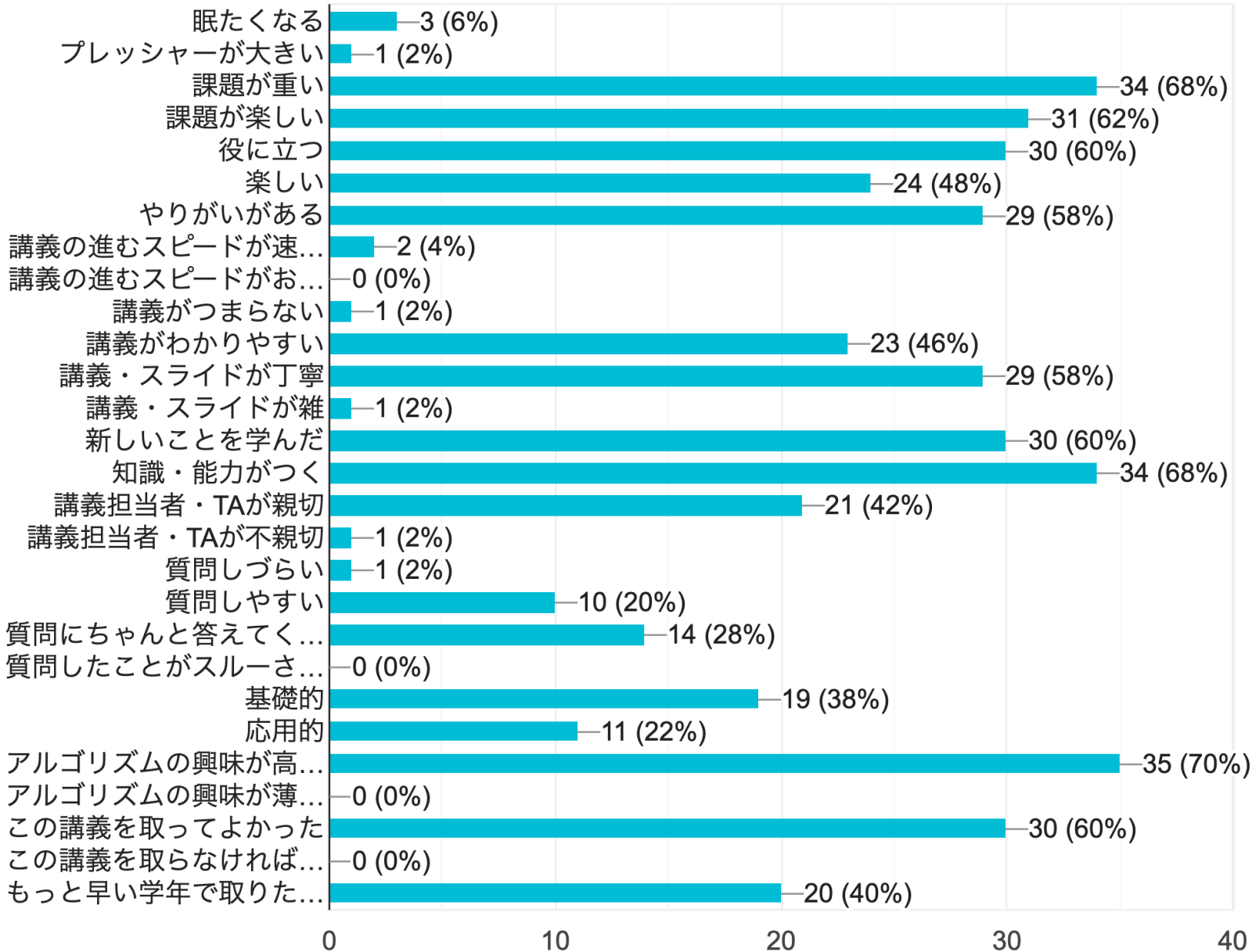
52 responses



2021年度  
アンケート結果

この講義での体験を他の人に伝えるときに使うで...う言葉を選んでください（当てはまるもの全て）

50 responses



2022年度  
アンケート結果

# 講義体制

講義担当者：矢谷 浩司

TA：大神 卓也，香取 浩紀，廣澤 佑亮，矢野 祥睦

ホームページ：<https://iis-lab.org/algorithms>

メールアドレス：[algorithms@iis-lab.org](mailto:algorithms@iis-lab.org)（矢谷 & TA）

基本は講義のslackにてコミュニケーションをお願いします。

# 受講に必要なデバイス, サービス

PC

OSは問わない

ブラウザとしてChromeかFirefoxを推奨

インターネット環境

本学学生：ECCS/utacアカウント

メタバース工学部生：Googleアカウント

# 講義の前提条件

使用言語：Python3

コーディング環境：track（後で説明します）

学科PCか自前のPCを持参し，ネットワークに接続できる状態にしておいてください。

ローカル環境でもpythonを実行できるようにしておいてもらえるとよいです（必須ではありません）。

# 講義の前提条件

プログラミング，pythonの基礎的内容は身につけているものとしします。

変数の型，配列，条件分岐，ループ，関数，再帰，入出力などなど。

このあたりが不安な人は今のうちに自習をお願いします。

# 講義

スライドを使って説明します。スライドは授業のホームページにもアップロードしておきます。

授業開始時刻5分前くらいから、Youtube Liveでの配信を開始します。

終了後も後から視聴できるようにしておきますので、見逃した方はお時間のある時に自習してください。



# 講義

## **本学学生：**

241講義室にて原則対面で受講.

## **メタバース工学部生：**

Youtube Liveにてオンラインで受講.

特段の事情（新型コロナウイルス感染症高リスク者等）によりオンラインで受講を希望する本学学生は別途相談してください.

# 講義の録画

## **本学学生：**

第1回と第13回のみ，講義終了後直後から閲覧可能。  
それ以外の回は，同週に行われたコードチャレンジの終了後（およそ1週間後）に閲覧可能。

## **メタバース工学部生：**

Youtubeで授業終了後から閲覧可能。

# Q&A

講義のslack (EEIC-Algorithms2023) に皆さんを招待しています。

各講義回チャンネルを用意していますので、そちらでコメントをお願いします。

/anonymousと打つと，無記名で投稿できます。

**教室に来ている方もslackでご質問ください。**

## ▼ Channels

# class01

# class02

# class03

# class04

# class05

# class06

# class07

# class08

# class09

# class10

# class11

# class12

# class13

# Slackのテストを兼ねて

#class01に投稿をお願いします。 /anonymousをつかって投稿してもらってもOKです。

本学学生さん：

今みなさんが特にハマっていること（食べ物，飲み物，アクティビティ，趣味，サークルなんでも）を教えてください！

メタバース工学部生さん：

御社が「他に負けないぞ！」と思うことを教えてください！

# 成績評価（本学学生）

**コードチャレンジ基本課題**（全24課題）：34点

**選択課題**（どちらかを選択，両方やった場合はより得点の高い方を成績に利用）：33点

コードチャレンジExtra課題（全11課題）

レポート課題（全1課題）

**期末試験**：33点

ただし，期末試験を放棄（出席しなかった）場合，この講義の成績は「未受験」という扱いになります。

# 成績評価（メタバース工学部生）

コードチャレンジ基本課題（全24課題，34点満点）の提出があり，以下の条件を全て満たす場合，「修了」となります。

総得点が17点以上。

0点の提出（未着手のままの提出，着手したが0点の提出，どちらの場合でも）が5回以下。

Omnicampusでの出席アンケートの提出が8回以上。

# 成績評価（メタバース工学部生）

Extra課題は提出しなくても成績に対するペナルティはありませんが，提出された場合は0.5を乗じた上で総得点に加算することとします．

例）基本課題で15点， Extra課題で10点を取った場合，  
→  $15 + 0.5 * 10 = 20$ となり， 修了基準を満たします．

# コードチャレンジ

2種類あります.

基本課題

Extra課題

コーディングする時間を定期的に作っていただくため、毎週お届けします.

課題の配信は終了15分くらい前、締切は次の週の火曜日の24:00です.



# 基本課題：全員できてほしい課題

必修課題ですので，全員提出してください。

主に，講義内で紹介された重要なアルゴリズムの実装や変更を行うものです。

授業の内容を追えばできるようになっていきます。

平均的には1～1.5時間程度で完成させられるくらいのボリューム感。

# Extra課題：頑張ればできる腕試し的課題

選択課題の1つ.

授業の内容を踏まえた発展的な課題で、平均的には6時間程度で完成させられるボリューム感.

ちょっと競技プログラミング的なノリを含んでいます.

コーディング能力をさらに高めたい人はぜひチャレンジしてみてください.

# コードチャレンジ

**締切は直近の火曜日の24:00ですが、「土日も課題に取り組み」というメッセージではございません。** 😊💧

お時間のある時を自由に選んで、できる限り課題に取り組みることのできる環境を提供するための設定ですので、ご理解ください。

土日に取り組みことを妨げることはありませんが、しっかりリフレッシュする時間を取ることも意識し、自分の生活・学習リズムを作ってもらえると嬉しいです。

# レポート課題

もう1つの選択課題.

既に掲載していますので、ご参照ください.

コーディングに自信がない人でも、内容をしっかり理解して取り組めば、良い点に繋がります.

# Extra課題とレポート課題の採点

どちらが特別に不利にならないように平均点，点数分布がある程度近くなるように，レポート課題の採点は調整します。

ただし，本質的に全く違う課題であり，提出数も大きく違うことが予想されますので，全く同じ得点分布になることは難しいとご承知おきください。

# 期末試験

現時点では，講義室にて実施する予定です。

70～80分程度の試験を予定しています。

ただし，今後の状況によって変更することもあり得ますので，授業中，slackでのアナウンスに従ってください。

# 講義内容

#1: イントロダクション, 計算量

#2: 累積和, 整数関連

#3: データ構造

#4: 探索 (サーチ)

#5: 整列 (ソート)

#6: 文字列照合

#7: 動的計画法1

#8: 動的計画法2

# 講義内容

#9: BFS, DFS

#10: グラフアルゴリズム1 (最短経路問題)

#11: グラフアルゴリズム2 (最小全域木, トポロジカルソート)

#12: グラフアルゴリズム3 (最大流問題, 最小費用流問題, 二部グラフのマッチング問題)

#13: 「難しい問題」とは, さいごに



# 出席（本学学生）

取りません。 😊

授業に出席したり，講義の配信や録画を視聴したりすることは必須ではありませんが，頑張ってお届けしますので，見ていただけると嬉しいです。

励ましのお便りもお待ちしております！

出席せずとも，課題には積極的に取り組んでください。

# 出席（メタバース工学部生）

別途、メタバース工学部より出席管理（Omnicampus）がありますので、ご対応をお願いします。

Omnicampusに関する質問はslackの#metaverse-onlyにてお願いいたします（メタバース工学部事務局の方がご対応いただけるはずです）。

# 病欠，公欠

以下の情報をメールにて，[algorithms@iis-lab.org](mailto:algorithms@iis-lab.org)に送ってください．認められた場合，課題提出期限をこちらが指定する日時まで延長します．

- 病欠，公欠を希望する授業日
- 病欠，公欠の理由
- 理由が正当なものであることを裏付けるもの（診断書，処方箋，出席する学会のプログラム等）

可能な限り事前に，事後でも速やかにお願いします．大きな怪我や病気などの例外を除き，病欠，公欠を希望する授業日から1週間以上を経過した場合は，認めないものとします．

# TAさん

講義の配信が見れない，trackが使えないなどのトラブルがあればslackで連絡ください。受講者が多い中で対応をお願いしていますので，その点ご理解ください。

みなさんのコードに対する個別のデバッグ等はサポートできかねますので，ご了承ください。

# Academic misconduct

この講義でも厳しく対処します。

剽窃，無断流用などが発覚した場合，その度合いに応じて以下のような処置が取られます。

- 提出された課題，試験を0点とする。
- 提出された課題，試験を0点とし，以降の提出を認めない。
- 全課題，試験に遡って0点とし，以降の提出を認めない。

# Academic misconduct

コードチャレンジでは皆さんに課題に対するコードを提出してもらいます。

Extra課題に関しては、提出されたコードに対して後日類似度チェック等を行い、明らかに類似したコードがあった場合は、その全てに対してペナルティを課します。

基本課題に関してはほぼ同じようなコードになるはずですので、このルールを直接は適用しませんが、チェックは行います。

# Academic misconduct

講義で使うシステムではコードをテストケースにかけた時点で、自動でバージョン管理します。

コピペ等を行なった場合には、不自然にコード行数が増えたり、テストケースにパスする回数が突然増えたりするので、その辺りもチェックします。

# Academic misconduct

本やWeb上の情報を参考にするのは構いません。ただし、そのまま使うことのないようにしてください。

友達同士で教え合うのも積極的にどうぞ。ただし、コードを直接見せ合うのではなく、考え方だけを共有するようにしてください。

「自分で手を動かす」ことをご自身で大切にしてください。



# Academic misconduct

生成AIや自然言語処理・人工知能サービスを利用してコードを生成し、その全部、もしくは一部を利用して回答した場合、**本講義においては即座に不正行為とみなし、今までに提出されたすべての課題を不採点、かつ以降の課題提出を認めないものとしてします。**

本やWebの情報をそのまま使うのと同じく、自分のために全くなりませんので、そのようなことに無駄に時間を使わないようにしていただければと思います。

# 外部リソースの活用

ただし、生成AIや自然言語処理・人工知能サービス、あるいはWebで調べた内容や説明、解説のうち、自分の理解に大きく役に立ったものがあれば、ぜひ講義の各回のチャンネルで共有してください。

**ただし、この場合も、コード例をそのまま貼り付けるなどはしないようにしてください。**

# Let's help each other!

課題等でわからないことがあれば、講義の各回のチャネルで共有してください。

私やTAさんのほか、学生の皆さんもできる範囲内で積極的にサポートしてもらえれば嬉しいです。

コメントを返すのを特に目立って行っている方には、別途成績で考慮したいと思います。

# 事前にお伝えしておきたいこと

矢谷が海外出張のため、4/26はオンラインのみとなる予定です。（直前で変更があるかもですが）

現地からリアルタイムでできるか、録画したもの（去年撮影したもの）をみていただく形になるかは改めてご連絡します。

# ご登録をお願いします！

本学学生：<https://forms.gle/eYiZRVPfxM1qx5A9>

メタバース工学部生：（メールでお願いしているフォーム）  
今すぐお願いします！

講義のslackやコードチャレンジで使うシステムへの招待を行う他，皆さんのバックグラウンドを把握するため，ご協力をお願いします。

単位の取得が必要な人は，別途UTASでの登録を忘れないようにお願いします。

では、早速！

# アルゴリズムとは（再掲）

あるタスクを達成するために設計された有限回の計算手順（ソート，サーチ，最適化などなど）。

アルゴリズムは正しい，あるいは「最適な」解を導くように設計されている。（ただし，最適と言っても，あくまでアルゴリズム内で設定された評価基準による）。

多くの場合は（時間コスト，メモリコスト，通信コストにおいて）効率的な計算手順を意味する。

# アルゴリズムの最低条件

有限実行時間で必ず止まる（停止性）。

止まった時，正しい結果が得られる。



# アルゴリズムの最低条件

有限実行時間で必ず止まる（停止性）。

止まった時，正しい結果が得られる。

まあ，そりゃそうじゃないと困りますよね。😅

気にしたいのは「いつ止まる」のか。

# 計算量

与えられたアルゴリズムがどの程度の時間的・メモリの  
コストで実行できるかの目安.

具体的な時間やメモリ量を表すものではなく，必要な  
コストを半定量的に表す指標.

大雑把な見積もり， という感じ.

# 計算量

与えられたアルゴリズムがどの程度の時間的・メモリのコストで実行できるかの目安.

具体的な時間やメモリ量を表すものではなく，必要なコストを半定量的に表す指標.

$O(n)$  とか  $O(n^2)$  という形で表す. (「オーダー $n$ 」というふうに読む.) ビックオー記法と呼ばれる.

# 時間計算量の例（線形探索）

ある配列の中から，別に与えられた値に一致する要素を探し出す．

配列の先頭から順にチェックして，一致する要素があればその時のindexを返す． ない場合は-1を返す．

例)

入力：[8, 3, 4, 1, 6, 9, 2]で6の場所を探す

出力：4

# 時間計算量の例（線形探索）

```
def search(sequence, key):  
    i = 0  
    while i < len(sequence):  
        if sequence[i] == key:  
            return i  
        i += 1  
    return -1
```

# 時間計算量の例（線形探索）

```
def search(sequence, key):  
    i = 0  
    while i < len(sequence):  
        if sequence[i] == key:  
            return i  
        i += 1  
    return -1
```

平均的な実行回数

1回

$n/2$ 回

$n/2$ 回

1回

$n/2$ 回

(1回)

# 時間計算量の例（線形探索）

```
def search(sequence, key):  
    i = 0  
    while i < len(sequence):  
        if sequence[i] == key:  
            return i  
        i += 1  
    return -1
```

$O(n)$

一番支配的な項のみで計算量を表す。

# 計算量

入力が $O(n)$ 規模である想定（ $n$ 個の要素の配列とか）。

平均的な場合と最悪の場合で計算量が変わることもある。

両方の場合で分けてたり，最悪の場合だけ考えたり，  
とその時々で違う。

表されている計算量がどんなケースを考えているもの  
なのか，正しく理解してから前に進んでください。



# オーダーの感覚

$$O(1)$$

$$O(\log n)$$

$$O(n)$$

$$O(n \log n)$$

$$O(nm)$$

$$O(n^2)$$

$$O(2^n)$$

$$O(n!)$$

(データの大きさによります)

# オーダーの感覚

(データの大きさによります)

$$O(1)$$



$$O(\log n)$$

$$O(n)$$

$$O(n \log n)$$

$$O(nm)$$

$$O(n^2)$$

$$O(2^n)$$

$$O(n!)$$

# オーダーの感覚

$$O(1)$$



$$O(\log n)$$



$$O(n)$$

$$O(n \log n)$$

$$O(nm)$$

$$O(n^2)$$

$$O(2^n)$$

$$O(n!)$$

(データの大きさによります)

# オーダーの感覚

$$O(1)$$



$$O(\log n)$$



$$O(n)$$



$$O(n \log n)$$

$$O(nm)$$

$$O(n^2)$$

$$O(2^n)$$

$$O(n!)$$

(データの大きさによります)

# オーダーの感覚

$$O(1)$$



$$O(\log n)$$



$$O(n)$$



$$O(n \log n)$$



$$O(nm)$$

$$O(n^2)$$

$$O(2^n)$$

$$O(n!)$$

(データの大きさによります)

# オーダーの感覚

$$O(1)$$



$$O(\log n)$$



$$O(n)$$



$$O(n \log n)$$



$$O(nm)$$



$$O(n^2)$$

$$O(2^n)$$

$$O(n!)$$

(データの大きさによります)

# オーダーの感覚

$O(1)$



$O(\log n)$



$O(n)$



$O(n \log n)$



$O(nm)$



$O(n^2)$



$O(2^n)$

$O(n!)$

(データの大きさによります)

# オーダーの感覚

$$O(1)$$



$$O(\log n)$$



$$O(n)$$



$$O(n \log n)$$



$$O(nm)$$



$$O(n^2)$$



$$O(2^n)$$



$$O(n!)$$

(データの大きさによります)



# オーダーの感覚

$$O(1)$$



$$O(\log n)$$



$$O(n)$$



$$O(n \log n)$$



$$O(nm)$$



$$O(n^2)$$



$$O(2^n)$$



$$O(n!)$$



(データの大きさによります)

オーダーの感覚

$$O(n^2)$$

$$O(n \log n)$$

$$O(n)$$

# オーダーの感覚

$$O(n^2)$$

$$n = 10^6$$

$$O(n \log n)$$

$10^8$  ステップ/秒

$$O(n)$$

# オーダーの感覚

$$O(n^2)$$

2.7時間

$$n = 10^6$$

$$O(n \log n)$$

200ミリ秒

$10^8$  ステップ/秒

$$O(n)$$

10ミリ秒

# ビッグオー表記

$T(n) = O(f(n))$ という形で表す.

十分大きな $n$ に対して、関数 $T(n)$ が、 $f(n)$ に比例、もしくはそれより小さいことを表す.

漸近的上界, ともいう.

この漸近的上界はいくらでも存在するが、できる限り精度良く & 簡潔に表現しているものを通常は採用する.

# ビッグオー表記

例えば,  $T(n) = n^2 + 4n + 100$ の場合, 以下はどれも間違っているわけではない.

$$O(n^2), O(n^2 + n), O(n^3), O(n^{100}), O(2^n)$$

ただし, この場合 $O(n^2)$ が最も簡潔に精度良く $T(n)$ を表現しているので, これを採用する.

この授業や多くの教科書でも以上のような前提・約束でビッグオー表記による説明を行っています.

この授業の試験においても同様.

# 空間計算量（領域計算量）

単純には、メモリの消費量。

IoTデバイスのようなメモリが限られる環境や、超大規模なデータを処理する場合などにはよく考える必要あり。

時間計算量とトレードオフになることもある。

アルゴリズムの授業では時間計算量の方が重要視される（ことが多いと思われる）。

# そのほかの計算量・計算コスト

通信コストなど.

Computational offloading (モバイルやウェアラブル端末において重い処理をクラウドなどに投げてしまう) などの場合には重要.



# 時間計算量を体験しよう！

以下のタスクを解くようなアルゴリズムを考えよう。

「ランダムな整数が格納されている長さ $N$ の配列の中で、 $M$ 個の隣接する要素の和が最大となる部分を1つ求めよ。」

入力：配列（長さ $N$ ， $0$ 以上の整数）と $M$

出力： $M$ 個の隣接する要素の和の最大値と，その $M$ 個の部分列の一番最初のindex.

# 時間計算量を体験しよう！

例) 配列[1, 1, 3, 4, 2]で隣り合う3つの要素の和が最大になるものはどれか？

[1, 1, 3, 4, 2]  $\rightarrow$  5

[1, 1, 3, 4, 2]  $\rightarrow$  8

[1, 1, 3, 4, 2]  $\rightarrow$  9で、これが最大.

# ナイーブな考え方

配列の1番目からM番目までを足し合わせ、その値を最大値として記録。indexは0を記録。

# ナイーブな考え方

配列の1番目からM番目までを足し合わせ、その値を最大値として記録。和が最大となる場所を保存する変数indexには0を入れておく。

次に、配列の2番目からM+1番目までを足し合わせ、この値が今の最大値より大きければ、最大値とindexを更新。

# ナイーブな考え方


配列の1番目からM番目までを足し合わせ、その値を最大値として記録。和が最大となる場所を保存する変数indexには0を入れておく。

次に、配列の2番目からM+1番目までを足し合わせ、この値が今の最大値より大きければ、最大値とindexを更新。

以降、同じ処理をN-M+1番目からN番目の要素の部分 and をチェックするまで繰り返す。（部分和が同じになる場合、indexの若い方を優先する。）

実際にやってみると, , ,

$N=100,000$ ,  $M=1,000$ くらいにすると, (私のそんなに非力じゃないマシンでも) 18秒くらいかかる.

おせー. . . 

# 効率的なアルゴリズムの着眼点

無駄を無くそう！

特に、**何度も同じことをやっているのを削減しよう。**

少し深くみてみよう.

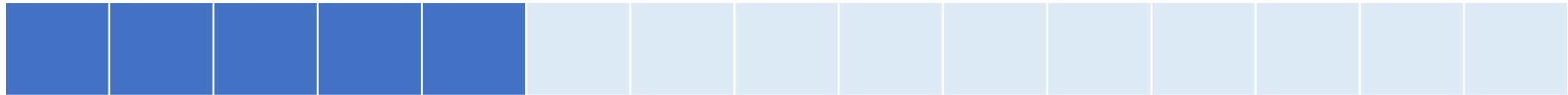
ある配列に対して, 隣り合う5つの要素の和に対して, 最大になるものを求めることを考える.





# ナイーブな方法の無駄はどこに？

1回目の計算：



# ナイーブな方法の無駄はどこに？

1回目の計算：



2回目の計算：



# ナイーブな方法の無駄はどこに？

1回目の計算：



2回目の計算：



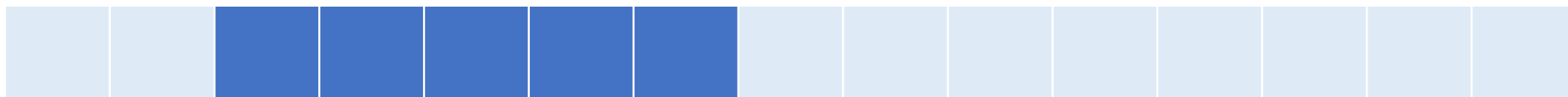
オレンジとのところは全く一緒！

# ナイーブな方法の無駄はどこに？

2回目の計算：



3回目の計算：

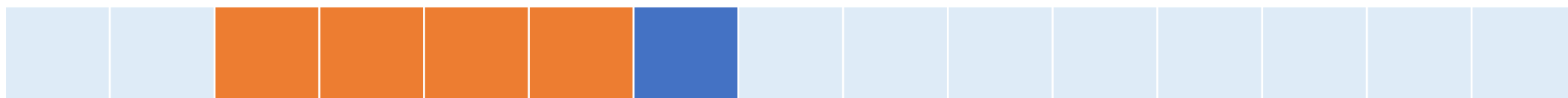


# ナイーブな方法の無駄はどこに？

2回目の計算：



3回目の計算：



オレンジとのところは全く一緒！

# 非効率な部分が見つかった！

毎回の計算で重複する部分があるが、それを毎回再計算してしまっている。

配列の要素に対する更新はないので、毎回計算する必要はないはず。

# 非効率な部分が見つかった！

毎回の計算で重複する部分があるが、それを毎回再計算してしまっている。

配列の要素に対する更新はないので、毎回計算する必要はないはず。

**差分だけうまく計算するにはどうすれば良い？**

今日の課題ではこれを考えてみてください！ 😊

差分だけうまく計算する方式にすると,

実際に実装したものでテストしてみると,  $N=100,000$ ,  
 $M=1,000$ くらいでも, (さっきと同じマシンで) 40ミリ秒  
くらいで終わる!

やったー! 😊



# コードチャレンジ：基本課題#1-b [1点]

スライドの説明を踏まえて、「ランダムな整数が格納されている長さ $N$ の配列の中で、 $m$ 個の隣接する要素の和が最大となる部分を1つ求めよ。」の問題を効率的に解くプログラムを実装してください。

# 実装できたら

自分のローカル環境で、配列の長さや $m$ の値を適当に変化させてみて、計算が完了するまでにどのくらい時間がかかるかを試してみてください。（例えば、配列の長さを10万くらいにするとどうでしょうか？）

ナイーブな方法も実装して比べてみてください。どのくらい実行時間が違いますか？

このアルゴリズムの計算量がいくらか、考えてみてください。

# ちなみに

ランダムに非負整数が並んだ配列は以下のコードで作れます。

----

```
import random
```

```
def RandomIntSeq(length):
```

```
    seq = random.sample(list(range(0, length)), k=length)
```

```
    return seq
```

# コードチャレンジ：基本課題#1-a [1点]

「RPGのショップ」

Pythonにおける標準入出力と明示的な型変換を扱う問題.

今後のコードチャレンジにおいてもよく使うと思いますので、今のうちに標準入出力と明示的な型変換ができるようになっておいてください.

# 開始時に入っている初期コードについて

このコードはtrackで自動的に設定されているものであり、この講義には関係ありません。

この初期コードを使用する必要はありません。（勿論使用した上で解答してもらっても構いません）。

# 入出力について

コードチャレンジでは標準入力により与えられるデータを受け取り，標準出力で計算結果を表示する必要があります。

# 出力

標準出力はprintで行うのが最もシンプルかと思います。  
(sys.stdoutなどを使ってもらっても勿論OK.)

printを使って出力すると自動的に末尾に改行が入ります。

# print実行例

コード

```
x = 1
```

```
y = [1, 2, 3]
```

```
print(x)
```

```
print(y)
```

```
print('string')
```

出力結果

```
1
```

```
[1, 2, 3]
```

```
string
```



# 入力

標準入力はinputで行うのが最もシンプルかと思います。  
(sys.stdinなどを使ってもらっても勿論OK.)

`a = input()` とすると、1行分読んでaに代入します。

**ただし、文字列として読み込まれることに注意！**

# input実行例

コード

```
a = input()
```

```
b = input()
```

```
print(a+b)
```

入力

1

2

出力結果

12

# 入力

必要に応じて型変換を行ってください。

例えば、int型に変えるためには、

```
int(input())
```

とすればよいです。

その他の型変換に関しては各自で調べてみてください。

# 入力

複数の値が1行に並んでいる場合はどうする？

例) 300 20

単にinput()しただけでは、「300 20」という文字列になってしまう...

# 入力

input()で取り込んだ後、スペースで分割し、さらにint型に変換する必要がある。

真面目にやると、

```
a = input()
```

```
b = a.split()
```

```
M = int(b[0])
```

```
L = int(b[1])
```

# 入力

map関数を使うとより簡潔に！

map関数：配列（リスト）などの各要素全てに対して指定する関数による操作を一括で適用する。

先ほどの処理は、

```
M, L = map(int, input().split())
```

と1行で記述できます。

# 入力

複数の値が複数行並んでいる場合はどうする？

例) 4 8 1  
10 20 30  
100 0 1000  
...

`map(int, input().split())`をループさせて読み込んでいく。  
何回ループを回すべきかは事前に与えられるはず。

# 入力

基本課題1-aでは文字列と数字が入り混じっているので、個別に扱う必要があります。

```
sword 400 10  
potion 50 1  
spear 250 25  
stick 70 1
```

```
item, *values = input().split()  
price, level = map(int, values)
```

みたいにするとういいます。



# コードチャレンジ：Extra課題#1

今日はExtra課題はありません。

コードチャレンジで使うシステムに慣れておいてください。

やってみよう！

ここから、コードチャレンジで使うシステムの説明.

システムのwalkthroughなどを残りの時間で行います.

コードチャレンジにおける  
trackの利用方法

trackの概要

# trackとは

Givery社が提供しているプログラミング学習・試験プラットフォーム



[私たちの想い](#) [プロダクト・サービス](#) [導入事例](#) [お役立ち情報](#) [イベント・ニュース](#)

[資料一覧](#) →



# trackとは

エンジニア採用のための試験として使われていることが多い。

今回，特別にGivery様のご協力を得て，この授業のためにシステムを利用させていただくことになりました！👷

# trackとは

左側にコーディング問題、右側にコーディング環境が表示され、ブラウザ上でコーディングを行うことができる。

右下では書いたコードに対するテストが実行され結果を確認できる。

The screenshot displays a coding interface for a problem titled "Rectangle" (difficulty 1-3). The problem description asks for a program to calculate the area and perimeter of a rectangle with sides  $a$  cm and  $b$  cm. The input is two integers  $a$  and  $b$  separated by a space, and the output is the area and perimeter also separated by a space. Constraints include integer inputs and  $1 \leq a, b \leq 100$ . A sample input of "3 5" is provided. The code editor on the right shows a Python script that reads the input, splits it into  $a$  and  $b$ , and prints the area and perimeter. A "CLEAR" button is visible below the code editor. The test results section shows two successful tests, with a summary indicating 2 tests passed and 0 failed. A "Preview mode" button is located in the bottom right corner.

**Rectangle** 1-3

たて  $a$  cm よこ  $b$  cm の長方形の面積と周の長さを求めるプログラムを作成して下さい。

**Input**

$a$  と  $b$  が 1 つの空白で区切られて与えられます。

**Output**

面積と周の長さを 1 つの空白で区切って 1 行に出力して下さい。

**Constraints**

- 全ての入力は整数
- $1 \leq a, b \leq 100$

**Sample Input**

```
3 5
```

```
1 import sys
2
3 if __name__ == "__main__":
4     a, b = map(int, input().split())
5     print(a * b, 2 * (a + b))
6
```

✓ CLEAR Enterキーを押して次に進もう

Ready!

✓ test1

✓ test2

# tests 2

# pass 2

# fail 0

Preview mode

# trackの特徴

オンラインエディタ上でコードを書くことができるので手元での環境構築が一切いらない。

幅広いプログラミング言語に対応（今回の講義ではPython3に限定していますが）。

その場でテストケースを実行することが出来、自分のコードに対するフィードバックがすぐに得られる。



本講義でのtrackの利用方法

# 本講義でのtrackの利用方法

講義があるたびに演習問題を計2～3題ずつ配信予定。

- 基本課題
- Extra課題

毎回の講義パート終了後， track上での課題を公開します。

track上では， 「受験」 「試験」 などという単語が出てきますが， 試験ではないので安心してください。

# 課題を行う手順

<https://eeic-algorithms.train.tracks.run/auth/login>

にアクセスしてください。

アカウント名，初期パスワードは以下のように設定されています。

- 本学学生

- アカウント名：ECCSアカウントのメールアドレス
- 初期パスワード：入力していただいた学生証番号

- メタバース工学部生

- アカウント名：ご登録時に使用したGoogleアカウントのメールアドレス
- 初期パスワード：（別途メールにてご連絡しております。）

# 課題を行う手順

ページが表示されない等の場合、ブラウザを変更するか、adblock等のプラグインを無効化してみてください。



# 課題を行う手順

右下にある「受験環境を確認する」を押し、すべての項目で問題なしになることを事前に確認してください。

者としてログイン

者としてログイン

 受験環境を確認する

# 課題を行う手順

この講義のクラスをクリックすると課題一覧ページに飛びます。

The screenshot shows the user interface for the 'EEIC-Algorithms' course. On the left is a dark green sidebar with navigation icons for 'Training', 'Dashboard', 'Class', 'Discussion', and 'Notes'. The main content area has a header with the course name and a user profile for 'Koji (test)'. Below the header, there's a breadcrumb trail 'ダッシュボード > クラス'. The main section features a large card for the class 'アルゴリズム2023' (2023年度アルゴリズム) with a '2時間' estimated lecture time and a progress bar showing '0%' completion. Below this is a 'マテリアル' (Materials) section listing two tasks: '基本課題#01-a' and '基本課題#01-b'. Each task card includes a category 'アルゴリズム', an infinity symbol for duration, a submission deadline of '2023年3月15日 00:00', and buttons for '未着手' (Not started) and '開始' (Start).

# 課題を行う手順 - 問題一覧ページ

Pythonコードを書くエディタ

The screenshot shows a web-based coding environment. On the left, there is a problem description for 'RPGのショップ' (RPG Shop). The description includes constraints like memory limit (512 MB) and execution time (2000 ms). It explains that the player can buy items from a shop, with item names  $Name_i$  and prices  $Price_i$ . The player's level  $Level_i$  determines which items they can buy. The task is to implement a CLI that reads input and outputs the list of purchasable items.

On the right, there is a code editor with a file named 'main.py'. The code is a Python script that reads lines from standard input and prints them in a specific format. The code is as follows:

```
1 import sys
2
3 def main(lines):
4     # このコードは標準入力と標準出力を用いたサンプルコードです。
5     # このコードは好きなように編集・削除してもらって構いません。
6     # ---
7     # This is a sample code to use stdin and stdout.
8     # Edit and remove this code as you like.
9
10    for i, v in enumerate(lines):
11        print("line[{0}]: {1}".format(i, v))
12
13 if __name__ == '__main__':
14     lines = []
15     for l in sys.stdin:
16         lines.append(l.rstrip('\r\n'))
17     main(lines)
18
```

Below the code editor, there is a terminal window showing the output of the program. The input is 'echo "Hello track!"' and the output is 'Hello track!'. The terminal is labeled 'テスト出力' (Test Output) and has a 'Ready!' status.

問題文および入力データのサンプル

ジャッジ結果が表示されるコンソール

# 課題を行う手順 – 問題一覧ページ

気をつけてほしいこと

与えられた**基本課題**においては、入出力や問題文で指示がある場合を除き、実装すべき処理において標準ライブラリや問題で指定されていない外部の関数やライブラリ等を使用しないでください。 そのような提出物はテストケースに合格していても、採点されませんで注意してください。

Extra課題に関しては、必要に応じて標準ライブラリや外部の関数などを使っても構いません。ただし、必要なものをすべて提出物の中に組み入れるようにしてください。また、その場合でもライブラリに処理の大部分を依存してしまっているコードである場合には得点が無効となります。メインの実装を自分であることが課題の本質である、とご理解ください。



# 課題を行う手順－問題一覧ページ

## 気をつけてほしいこと

問題文をよく読んでから課題に取り組んでください。

## 課題によってはコードが一部予め与えられているものがあります。

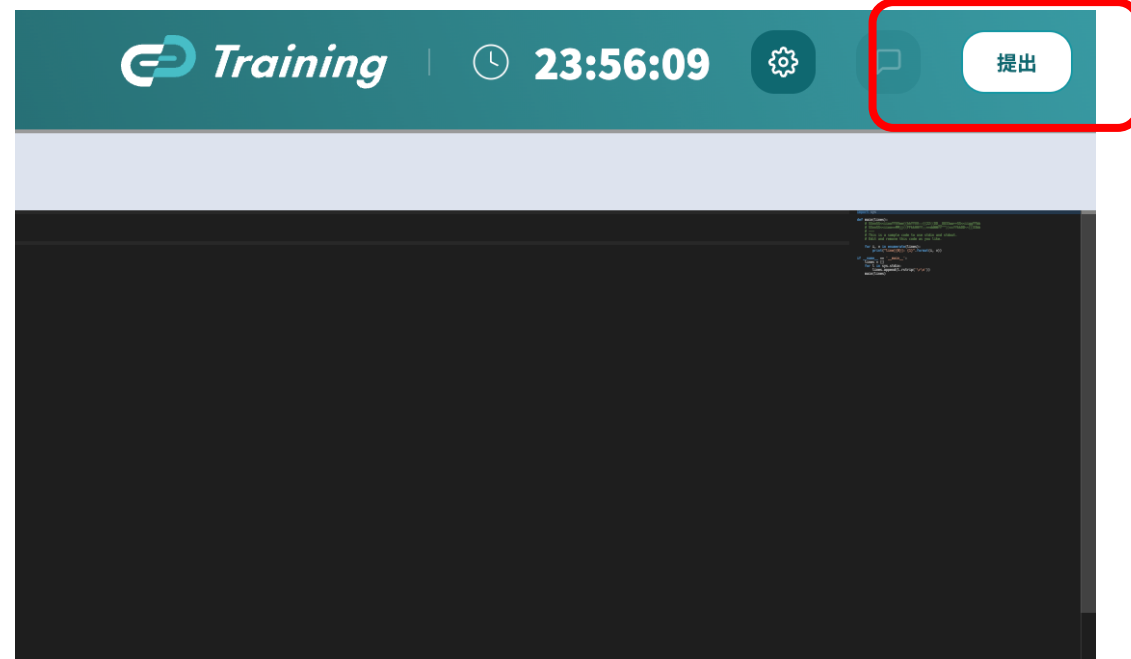
(track上でデフォルトで与えられるコードと違う場合があります。)  
この場合、問題文内で指示がありますので、それに従ってください。  
従っていない提出物は採点されないことがあります。

# 課題を行う手順 – 問題一覧ページ

気をつけてほしいこと

問題ページ右上の 「提出」を押して一度提出してしまうと再編集ができなくなります。

作業途中の時は右下の「テストを実行」を押すと、その時点でのコードが自動的に保存されるので、それを活用してください。



# 課題を行う手順 – コーディング環境について

本講義ではPython3に限定しています。

基本的に標準ライブラリしか含まれていません。状況によって、numpyなど一部のライブラリは追加するかもしれません。

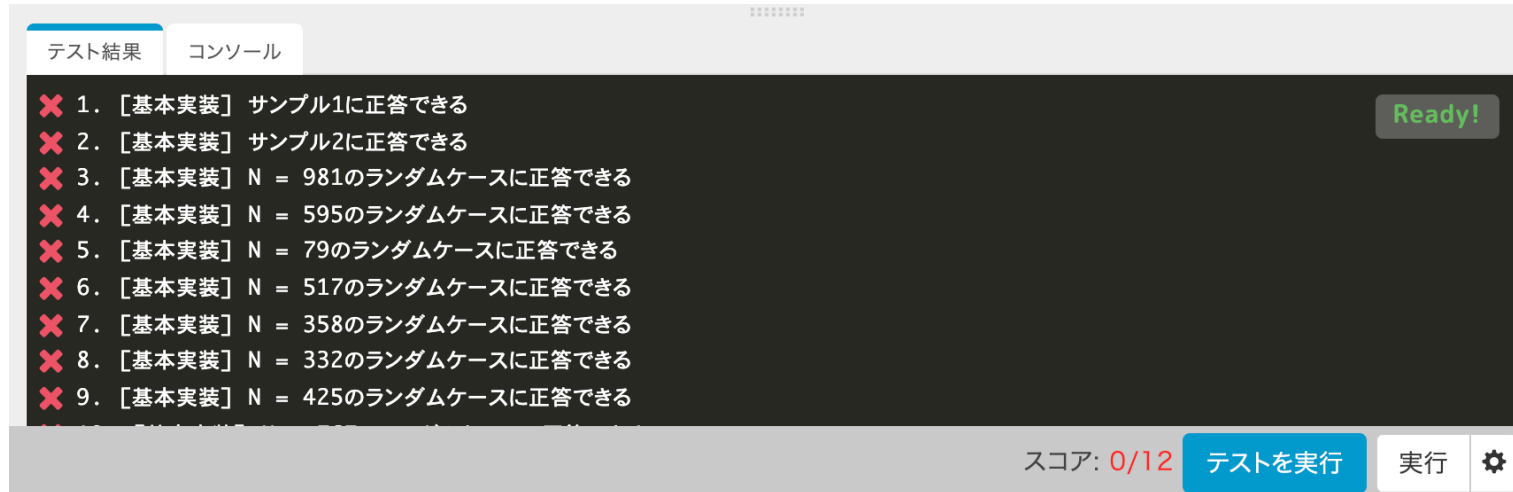
実装・デバッグをできる限りtrack上で行ってください。剽窃等の疑いが生じたときにtrack上でコードの更新履歴を確認することがあります。

# 課題を行う手順 - ジャッジについて

画面の右下の「**テストを実行**」(または⌘ + Sなど)を押すと用意されたテストケースが実行されます。

右下の「実行」ボタンはこちらでテストケースを自由に設定してコードを実行することができます。

「**テスト結果**」のタブにはそれぞれのテストケースの合否が、「**コンソール**」のタブには具体的な出力・エラーなどが表示されます。



The screenshot shows a coding platform interface with two tabs: "テスト結果" (Test Results) and "コンソール" (Console). The "テスト結果" tab is active, displaying a list of 9 test cases, all marked as failed with a red 'X' icon. The console tab is also visible, showing a "Ready!" message. At the bottom right, there is a score display "スコア: 0/12", a blue button "テストを実行" (Run Tests), a grey button "実行" (Run), and a gear icon for settings.

Test Case	Result
1. [基本実装] サンプル1に正答できる	Failed
2. [基本実装] サンプル2に正答できる	Failed
3. [基本実装] N = 981のランダムケースに正答できる	Failed
4. [基本実装] N = 595のランダムケースに正答できる	Failed
5. [基本実装] N = 79のランダムケースに正答できる	Failed
6. [基本実装] N = 517のランダムケースに正答できる	Failed
7. [基本実装] N = 358のランダムケースに正答できる	Failed
8. [基本実装] N = 332のランダムケースに正答できる	Failed
9. [基本実装] N = 425のランダムケースに正答できる	Failed

スコア: 0/12    テストを実行    実行    ⚙️

# 課題を行う手順 - テストケースについて

各問題について15個程度テストケースが用意されており，通ったテストケースの数に比例して採点が行われます。

テストケースは全て皆さんも確認できるようになっています。

テスト結果    コンソール

- ✓ 1. [基本実装] サンプル1に正答できる
- ✓ 2. [基本実装] サンプル2に正答できる
- ✓ 3. [基本実装] N = 981のランダムケースに正答できる
- ✓ 4. [基本実装] N = 595のランダムケースに正答できる
- ✓ 5. [基本実装] N = 79のランダムケースに正答できる
- ✓ 6. [基本実装] N = 517のランダムケースに正答できる
- ✓ 7. [基本実装] N = 358のランダムケースに正答できる
- ✓ 8. [基本実装] N = 332のランダムケースに正答できる
- ✓ 9. [基本実装] N = 425のランダムケースに正答できる

Ready!

スコア: 12/12    テストを実行    実行    ⚙️

# 課題を行う手順 – メモリ制限・実行時間

ほぼ全ての問題で

メモリ上限: **512MB**

実行タイムアウト: **5000ms**

に設定する予定です。

あまりに実行時間がシビアな問題は一部を除いてありませんが、Pythonには処理が遅い書き方などがあるので注意してください。

# 課題を行う手順 – 解答の提出

それぞれの問題でもうこれ以上変更を加えなくて良い状態になったら、右上の「**提出**」をクリックしてください。

一度提出を完了するとコードの変更はできません。 再提出のリクエストは対応できないことがあるので、注意してください。

# 課題提出にあたっての注意点

学生さん同士で議論することは推奨します。ただし、コードを直接共有する、などを行わないようにしてください。

参考書、Webサイト等を適宜参考にしながら、課題に取り組んでもらっても構いません。

TAさんはtrack上のトラブル等には対応しますが、個々のコードのデバッグには手助けできませんので、ご承知おきください。

質問はしていただいてOKですが、課題の性質上答えられないこともあります。



# 課題提出にあたっての注意点

**生成AIによってコードを作り出し、その全部、もしくは一部を用いて課題提出を行なった場合、この講義では不正行為とみなします。**

重大な結果を招くこととなりますので、不正行為を行わず、実直に課題に取り組んでください。

# 課題提出にあたっての注意点

個々の課題（基本課題，Extra課題）自体には制限時間は設定されていませんが，提出期限は設定されていますので，勘違いしないように注意してください。

## マテリアル

 **基本課題#01-a**

> アルゴリズム ⌚ ∞

提出期限: 2023年3月15日 00:00

 **基本課題#01-b**

> アルゴリズム ⌚ ∞

提出期限: 2023年3月15日 00:00

# 課題提出にあたっての注意点

可能な限りtrack上で作業をしてください。 **残念ながら自動保存の機能はないので、こまめに保存してください。**

頻繁にテストケースを実行し、デバッグを行ってください。これにより提出間際でのミスを防ぐことができる他、コードが自動保存され、万が一提出作業を行えなくても、テストケース合格率が最も高い最新のコードを元に採点が行われます。

採点は提出物に対してのみ行われます。したがって、途中でどれだけミスをしていてもペナルティはありません。

# 課題提出にあたっての注意点

提出物に対しては後日類似度チェックを行います。以下のような提出物は該当するものすべてに対して採点を取り消します。

- 提出物間でほぼ同一のコード
- Webや参考書に記載されているものとほぼ同一であることが判明したコード
- Track上での課題取り組み時間（終了時刻 - 開始時刻）が極端に短い
- 与えられたテストケースに通るようにだけ設計されたコード
- 課題で指定されている条件を満たしていないコード
- その他、明らかに不正行為の証拠があるもの

悪質な場合にはより大きなペナルティになることがあります。十分気をつけてください。

# コードの返却

提出後も自分のコードをtrack上で見ることが可能です。

Extra課題に関しては、後日簡単な解説と正答例もtrack上で提供します。

また、自習のためにtrack上で再度自分のコードをテストすることも可能です。ただし、返却後のコードは再採点されません（返却後に点数が上がっても、その点数は成績には反映されません）。